L4 - Operators and More

January 16, 2020

1 Operators, Lifetime, Scope, and Simple Class

1.1 Important Operators

1.1.1 Boolean Operators

- `A && B`: $A \cdot B$; A and B
- `A || B`: $A + B$; A or B
- `!A`: $\overline{A}$; not A
- `A ^ B`: $A \oplus B$; the **exclusive-or** operator.

Note that the result of the all the Boolean operators will be either true or false.

[3]:
```java
// making a choice in Java

int amount = 4;

if ( amount > 5 )
{
    System.out.println("yup");
}
else
{
    System.out.println("lol nope");
}
```

```
lol nope
```

1.1.2 Comparison Operators

- `A > B`: A strictly greater than B
- `A < B`: A strictly less than B
- `A <= B`: A less than or equal to B
- `A >= B`: A greater than or equal to B
- `A == B`: A is equal to B
  - don't confuse this with assignment!

- `A != B`: A is not equal to B

1.1.3 Mathematical Operators

- `A + B`: A plus B, duh
- `A += B`: *compound assignment*: add A and B and assign the result into A.

1.2 Lifetime and Scope

1.2.1 Lifetime

The *lifetime* of a variable describes how long ht evariable continues to exist before it is destroyed.

The lifetime of a variable is *dynamic*; it is affected by the particular object that it is part of, which can be changed by the user's operation of the program.

1.2.2 Scope

Scope of a variable defines the section of source code that can access that variable. Scopes can be nested; a statement within a block with in a class, for example.

Scope is static - it is defined by the structure of the program.

Local Scope A *local variable* can be defined in a *local scope*, which will be the block in which it is defined. Their values are defined within the method (maybe a constant for a calculation). These have short lifetimes and exist only as long as the method that uses them. They can only be accessed from within the method.

Local variables do not need a visibility keyword. A useful example of a local value would be storing a return value before resetting it (which must be done before the return statement).

[ ]:
```java
// from Better Ticket Machine

public int refundBalance()
    {
        int amountToRefund;  // amountToRefund is a local variable
        amountToRefund = balance;
        balance = 0;
        return amountToRefund;
    }
```

1.3 LabClass Example

[15]:
```java
// a student class
public class Student
{
    private String name;    // student's name
    private String id;      // student ID
    private int credits;    // credits taken so far

    /**
    *  Create a new student - a constructor method with two arguments.
    */

    public Student( String fullName, String studentID )
    {
        name = fullName;
        id = studentID;
        credits = 0; // note we have no parameter for this!
    }

    /**
    *  Accessor and mutator methods
    */

    public String getName()
    {
        return name;
    }

    public void changeName( String newName )
    {
        name = newName;
    }

    public String getStudentID()
    {
        return id;
    }

    public void addCredits( int additionalPoints )
    {
        credits += additionalPoints;
    }

    public int getCredits()
    {
        return credits;
    }

    public String getLoginName()
    {
        return name.substring(0,4) + id.substring(0,3); // first 4 letters of␣
↪name and first 3 digits of ID
    }

    public void print()
    {
        System.out.println( name + ", student ID: " + id + ", credits: " +␣
↪credits );
    }
}
```

[16]:
```
Student Alice = new Student("Alice in Wonderland", "12345");
```

[17]:
```
Alice.getStudentID();
```

[17]:
```
12345
```

[23]:
```
Alice.changeName("Alice");
```

[24]:
```
Alice.getName()
```

[24]:
```
Alice
```

[25]:
```
Alice.getLoginName();
```

[25]:
```
Alic123
```